# Neural Sum Rate Maximization for AI-Native Wireless Networks: Alternating Direction Method of Multipliers Framework and Algorithm Unrolling

Siya Chen
siyachen4-c@my.cityu.edu.hk
City University of Hong Kong
Hong Kong, China

Chee Wei Tan
cheewei.tan@ntu.edu.sg
Nanyang Technological University
Nanyang Ave., Singapore

## ABSTRACT

In this paper, we introduce *Neural Sum Rate Maximization* to address nonconvex problems in maximizing sum rates with a total power constraint for downlink multiple access. We combine the optimization-theoretic methods and neural network-based algorithm unrolling techniques to map iterations onto trainable neural network layers, enabling development and deployment in the AI-native layer for future wireless networks. Our approach leverages mathematical structures in sum rate optimization, such as the alternating direction method of multipliers and the standard interference function framework for iterative algorithm design. By integrating algorithm unrolling techniques, our approach can learn from data and significantly enhance efficiency. Numerical experiments demonstrate the method's advantages in efficiency, performance, and interpretability, which are particularly beneficial for resource-constrained optimization in AI-native wireless networks.

## CCS CONCEPTS

• **Networks** → **Network performance evaluation**; **Network performance analysis**; • **Computing methodologies** → **Machine learning**.

## KEYWORDS

Sum rate maximization, Alternating direction method of multipliers, Majorization-minimization, Algorithm unrolling

**ACM Reference Format:**
Siya Chen and Chee Wei Tan. 2024. Neural Sum Rate Maximization for AI-Native Wireless Networks: Alternating Direction Method of Multipliers Framework and Algorithm Unrolling. In *Proceedings of 2nd International Workshop on Networked AI Systems (NetAISys '24).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3662004.3663552

## 1 INTRODUCTION

The sum rate maximization problem in wireless networks is crucial for enhancing efficiency, reducing interference, and improving user experience [4, 14, 18]. With next-generation networks accommodating diverse applications, efficiently maximizing sum rate becomes more challenging. Various optimization approaches have been developed over the years to balance complexity and achieve maximum throughput [4, 5, 14, 18, 20, 24, 25]. Methods such as branch-and-bound [25] and outer approximation [18] aim for globally optimal solutions but are computationally intensive, while others offer more efficient but sub-optimal solutions, such as successive convex approximation [4, 19] and the weighted minimum mean squared error (WMMSE) algorithm [5], and so on.

Recent machine learning advancements enable learning-based end-to-end optimization for optimal resource allocation strategies from observed data [1, 12, 21]. Moreover, next-generation wireless network protocols are expected to incorporate AI-native designs, making machine learning techniques an appealing option for wireless optimization [10, 15]. While recent research explores various machine learning techniques like deep neural networks (DNNs), graph neural networks (GNNs), and deep reinforcement learning (DRL) [8, 11, 16] for wireless optimization, it is unlikely that machine learning alone will entirely replace traditional optimization in communication systems due to the need for substantial training data and costs.

The integration of modern machine learning with traditional optimization approaches shows promise in enhancing effectiveness and efficiency for wireless optimization [1, 3, 10, 15, 21]. One of the most significant techniques for this integration is the algorithm unrolling technique [6, 7, 9]. This technique enables bridging optimization methods with deep learning [6, 7, 9]. Recently, many algorithm unrolling applications aim to maximize the sum rate, such as [13], with a focus on unrolling the WMMSE algorithm [5]. However, challenges arise in implementing the WMMSE algorithm as neural network layers due to complex operations such as matrix inversions and eigendecompositions, making it hard to eliminate the performance gap.

In this paper, we introduce *neural sum rate maximization*, a novel approach combining machine learning and optimization algorithms for sum rate maximization. Instead of directly unrolling the WMMSE algorithm, we first propose a fast majorization-minimization (MM) method [17] using the standard interference function framework [22] and alternating direction method of multipliers (ADMM) [2]. Then, we employ algorithm unrolling techniques to develop an intuitive and easier-to-train learning model. The contributions of our paper are as follows:

(1) We introduce the MM algorithm, based on ADMM and the standard interference function framework, to solve downlink


*NetAISys '24, June 3-7, 2024, Minato-ku, Tokyo, Japan*

ACM ISBN 979-8-4007-0661-5/24/06
https://doi.org/10.1145/3662004.3663552

sum rate maximization. The algorithm, though it may not always guarantee global optimality, forms a foundation for combining domain-specific techniques with machine learning approaches for sum rate maximization.

(2) We propose *neural sum rate maximization*, a method that merges machine learning and domain-specific techniques with algorithm unrolling to improve efficiency and performance. The model preserves the architectures of iterative algorithms, enhancing their intuitiveness, interpretability, and ease of training. By integrating neural networks, this approach can effectively learn from data and significantly enhance efficiency.

(3) We present numerical results demonstrating the superior effectiveness and efficiency of the downlink neural sum rate maximization approach over traditional optimization-based and other modern machine learning algorithms.

The rest paper is structured as follows: Section 2 presents the sum rate maximization problems and fast MM algorithms using the standard interference function framework and ADMM. Section 3 introduces algorithm unrolling techniques for developing the neural sum rate maximization approach. Section 4 showcases numerical simulations, and Section 5 concludes the paper.

# 2 THE WEIGHTED SUM RATE MAXIMIZATION PROBLEM

## 2.1 System Model and Problem Formulation

We consider a multi-access wireless system with $L$ transmitter-receiver pairs. Let $x_l \in \mathbb{C}$ denote the $l^{th}$ transmitted signal, and the received signal $y_l \in \mathbb{C}$ can be expressed as [18]

$$y_l = h_{ll}x_l + \sum_{j=1, j\neq l}^{L} h_{lj}x_j + z_l,$$

where $h_{ll} \in \mathbb{C}$ denotes the channel between the $l^{th}$ transceiver pair, $h_{lj} \in \mathbb{C}$ (for $l \neq j$) represents interference from the $j^{th}$ transmitter to the $l^{th}$ receiver, and $z_l$ represents independent and identically distributed additive complex Gaussian noise with variance $n_l/2$ on each real and imaginary component. The transmit power vector is denoted by $\mathbf{p} = [p_1, \ldots, p_L]^\top$, with a total power constraint on the transmitters such that $\sum_l p_l \leq \bar{P}$, where $\bar{P}$ is the maximum total power. Let $G = \left[G_{lj}\right]_{l,j=1}^{L} > \mathbf{0}_{L\times L}$ be the channel gain matrix, where $G_{lj} = \left|h_{lj}\right|^2$ represents the channel gain from the $j^{th}$ transmitter to the $l^{th}$ receiver. Additionally, define $\mathbf{n} = [n_1, \ldots, n_L]^\top > \mathbf{0}$ with $n_l$ as the noise power at the $l^{th}$ receiver. The Signal-to-Interference-plus-Noise Ratio (SINR) for a receiver is the ratio of the received signal power to the sum of interference signal power and additive noise power. Specifically, the SINR of the $l^{th}$ receiver can be calculated as [18]

$$\mathrm{SINR}_l(\mathbf{p}) = \frac{G_{ll}p_l}{\sum_{j=1, j\neq l}^{L} G_{lj}p_j + n_l}.$$

Our objective is to determine the power allocation vector $\mathbf{p}$ that maximizes the total transmitted data rate, i.e., to solve the problem of sum-rate maximization through power control, which is stated as follows [18]:

$$\begin{aligned} \text{maximize} \quad & \sum_{l=1}^{L} w_l \log(1 + \mathrm{SINR}_l(\mathbf{p})) \\ \text{subject to} \quad & \sum_{l=1}^{L} p_l \leq \bar{P}, \\ \text{variables:} \quad & \mathbf{p} \in \mathbb{R}_+^L, \end{aligned} \tag{1}$$

where $\mathbf{w} = [w_1, \ldots, w_L]^\top > \mathbf{0}$ is a given probability vector and $\bar{P}$ is a given maximum transmit power.

## 2.2 Algorithm for Sum Rate Maximization

Now we propose to solve (1) using the MM algorithm based on ADMM with the standard interference function framework. Firstly, we reformulate (1) as

$$\begin{aligned} \text{minimization} \quad & \sum_{l=1}^{L} w_l \log\Big(\sum_{j=1, j\neq l}^{L} G_{lj}p_j + n_l\Big) \\ & - \sum_{l=1}^{L} w_l \log\Big(\sum_{j=1}^{L} G_{lj}p_j + n_l\Big) \\ \text{subject to} \quad & \sum_{l=1}^{L} p_l \leq \bar{P}, \\ \text{variables:} \quad & \mathbf{p} \in \mathbb{R}_+^L. \end{aligned} \tag{2}$$

Note that (2) is still nonconvex. To tackle this problem, we start with the Jensen's inequality for the convex function $-\log x$, from which we have

$$\begin{aligned} -\log\Big(\sum_{j=1}^{L} G_{lj}p_j + n_l\Big) \leq & -\sum_{j=1}^{L} \alpha_{lj} \log p_j - \sum_{j=1}^{L} \alpha_{lj} \log \frac{G_{lj}}{\alpha_{lj}} \\ & -\alpha_{l,L+1} \log \frac{n_l}{\alpha_{l,L+1}}, \end{aligned} \tag{3}$$

where $\sum_{j=1}^{L+1} \alpha_{lj} = 1$ with $\alpha_{lj} \geq 0$ for all $l$ and $j$.

Then we aim to solve (2) by employing the MM algorithm. Denote $\mathbf{p}(k)$ as the power at the $k^{th}$ iteration. Then, the surrogate function of the objective function in (2) at $\mathbf{p}(k)$ is

$$\begin{aligned} S(\mathbf{p}|\mathbf{p}(k)) = & \sum_{l=1}^{L} w_l \log\Big(\sum_{j=1, j\neq l}^{L} G_{lj}p_j + n_l\Big) - \sum_{l=1}^{L}\sum_{j=1}^{L} w_l \alpha_{lj}(k) \log p_j \\ & -\sum_{l=1}^{L} w_l \left(\sum_{j=1}^{L} \alpha_{lj}(k) \log \frac{G_{lj}}{\alpha_{lj}(k)} + \alpha_{l,L+1}(k) \log \frac{n_l}{\alpha_{l,L+1}(k)}\right), \end{aligned} \tag{4}$$

where

$$\alpha_{lj}(k) = \begin{cases} \dfrac{G_{lj}p_j(k)}{\sum_{j=1}^{L} G_{lj}p_j(k) + n_l}, \text{ if } j \leq L \\ \dfrac{n_l}{\sum_{j=1}^{L} G_{lj}p_j(k) + n_l}, \text{ if } j = L+1. \end{cases} \tag{5}$$

Then we can approximate (2) at $\mathbf{p}(k)$ as

$$\begin{aligned} &\text{minimize} \sum_{l=1}^{L} w_l \log\Big(\sum_{j=1,j\neq l}^{L} G_{lj}p_j + n_l\Big) - \sum_{l=1}^{L}\sum_{j=1}^{L} w_l \alpha_{lj}(k) \log p_j \\ &\text{subject to} \quad \sum_{l=1}^{L} p_l \le \bar{P}, \\ &\text{variables:} \quad \mathbf{p} \in \mathbb{R}_+^L. \end{aligned} \tag{6}$$

Now we consider solving (6) by using the ADMM method [2]. By introducing auxiliary variables $\mathbf{z} \in \mathbb{R}_+^L$, (6) is equal to

$$\begin{aligned} &\text{minimize} \sum_{l=1}^{L} w_l \log\Big(\sum_{j=1,j\neq l}^{L} G_{lj}p_j + n_l\Big) - \sum_{l=1}^{L}\sum_{j=1}^{L} w_l \alpha_{lj}(k) \log p_j \\ &\qquad + g(\mathbf{z}) \\ &\text{subject to} \quad \mathbf{p} - \mathbf{z} = \mathbf{0}, \\ &\text{variables:} \quad \mathbf{p}, \mathbf{z} \in \mathbb{R}_+^L, \end{aligned} \tag{7}$$

where $g(\mathbf{z}) : \mathbb{R}_+^L \to \mathbb{R}$ denotes an indicator function of the set $\mathcal{Z} = \{\mathbf{z} : \sum_{l=1}^{L} z_l \le \bar{P}\}$, i.e.,

$$g(\mathbf{z}) = \begin{cases} 0, \text{ if } \sum_{l=1}^{L} z_l \le \bar{P}, \\ +\infty, \text{ otherwise .} \end{cases}$$

Then the partial augmented Lagrangian function of (7) is given as

$$\begin{aligned} \mathcal{L}(\mathbf{p},\mathbf{z},\mathbf{y}) = &\sum_{l=1}^{L} w_l \log\Big(\sum_{j=1,j\neq l}^{L} G_{lj}p_j + n_l\Big) - \sum_{l=1}^{L}\sum_{j=1}^{L} w_l \alpha_{lj}(k) \log p_j \\ &+ g(\mathbf{z}) + \mathbf{y}^\top(\mathbf{p}-\mathbf{z}) + \frac{\beta}{2}\|\mathbf{p}-\mathbf{z}\|_2^2, \end{aligned}$$

where $\mathbf{y}$ is the dual multipliers and $\beta$ is the penalty parameters. Using the scaled Lagrangian multiplier $\boldsymbol{\mu} = (1/\beta)\mathbf{y}$, the subproblems can be expressed as

$$\begin{cases} \mathbf{p}(t+1) = \text{argmin}_{\mathbf{p}} \sum_{l=1}^{L} w_l \log\Big(\sum_{j=1,j\neq l}^{L} G_{lj}p_j + n_l\Big) \\ \qquad\qquad - \sum_{l=1}^{L}\sum_{j=1}^{L} w_l \alpha_{lj}(k) \log p_j + \frac{\beta}{2}\|\mathbf{p}-\mathbf{z}(t)+\boldsymbol{\mu}(t)\|_2^2, \\ \mathbf{z}(t+1) = \text{argmin}_{\mathbf{z}}\ g(\mathbf{z}) + \frac{\beta}{2}\|\mathbf{p}(t+1)-\mathbf{z}+\boldsymbol{\mu}(t)\|_2^2, \\ \boldsymbol{\mu}(t+1) = \boldsymbol{\mu}(t) + \mathbf{p}(t+1) - \mathbf{z}(t+1). \end{cases} \tag{8}$$

For solving the subproblems for $\mathbf{p}(t+1)$ and $\mathbf{z}(t+1)$, we have the following lemmas.

Lemma 2.1. *The global minimizer of the subproblem for* $\mathbf{p}(t+1)$ *in* (8) *is the fixed point of the following iteration on* $\mathbf{p}(t,d)$ *indexed by* $d$*:*

$$p_l(t,d+1) = \frac{\mathbf{w}^\top \boldsymbol{\alpha}_l(k)}{\beta(p_l(t,d) - z_l(t) + \mu_l(t)) + \sum_{i=1,i\neq l}^{L} \frac{w_i G_{il}}{\sum_{j=1,j\neq i}^{L} G_{ij}p_j(t,d) + n_i}}. \tag{9}$$

Lemma 2.2. *The closed form of the global minimizer of subproblem for* $\mathbf{z}(t+1)$ *in* (8) *is*

$$z_l(t+1) = p_l(t+1) + \mu_l(t) - \frac{1}{L}\left(\sum_{l=1}^{L}(p_l(t+1)+\mu_l(t)) - \bar{P}\right)_+, \tag{10}$$

*where* $(x)_+$ *means* $(x)_+ = x$ *if* $x \ge 0$ *and* $(x)_+ = 0$ *otherwise.*

The proofs of Lemma 2.1 and 2.2 can be found in the Appendix. According to Lemma 2.1 and 2.2, the subproblems in the ADMM method can be solved efficiently. Then, by combining the MM algorithm and the ADMM method, we propose Algorithm 1 to solve (1). By selecting an appropriate initial point, this algorithm converges to the global optimal solution. While global optimality is not always guaranteed, Algorithm 1 sets the foundation for integrating machine learning techniques to solve sum rate maximization. Next, we enhance Algorithm 1 by integrating deep learning techniques through algorithm unrolling.

**Algorithm 1:** Sum Rate Maximization

**Input**: System parameters $\mathbf{G}$, $\mathbf{n}$, $\mathbf{w}$ and $\bar{P}$; Penalty $\beta$.
**Output**: Optiml power $\mathbf{p}^*$ for (1).
Randomly initialize $\mathbf{p}(0)$, $\mathbf{z}(0)$ and $\boldsymbol{\mu}(0)$.
**repeat**
  $\alpha_{lj}(k) = \dfrac{G_{lj}p_j(k)}{\sum_{j=1}^{L} G_{lj}p_j(k) + n_l}$; (11)
  **repeat**
    Update $p_l(t+1)$ to be the fixed point of the iteration (9) indexed by $d$;
    Update $z_l(t+1)$ with (10);
    Update $\mu_l(t+1)$ with $\mu_l(t+1) = \mu_l(t) + p_l(t+1) - z_l(t+1)$;
  **until** $\mathbf{p}(t), \mathbf{z}(t)$ and $\boldsymbol{\mu}(t)$ converge;
  $\mathbf{p}(k+1) \leftarrow \mathbf{p}(t)$;
**until** $\boldsymbol{\alpha}(k)$ and $\mathbf{p}(k)$ converge;

## 3 THE NEURAL SUM RATE MAXIMIZATION WITH ALGORITHM UNROLLING

This section introduces the neural sum rate maximization approach for addressing (1). Note that Algorithm 1 utilizes the MM algorithm to solve (1), with an outer loop updating $\boldsymbol{\alpha}(k)$ and inner loops solving the approximate convex subproblems using ADMM. Our approach leverages algorithm unrolling to transform all the loops into feed-forward structures with fixed-depth layers. The architecture is illustrated in Figure 1.

### 3.1 The Neural Sum Rate Maximization

In order to make (11) trainable, we introduce a neural network model. Given the high dimension of $\boldsymbol{\alpha}(k)$ for the outer loop in Algorithm 1, here we enable the learning model to learn $\mathbf{q}(k) = \mathbf{w}^T\boldsymbol{\alpha}(k)$ rather than directly learning $\boldsymbol{\alpha}(k)$. Then the $k^{th}$ outer loop can be represented as:

$$\mathbf{q}(k) = \boldsymbol{\psi}(\mathbf{p}(k), \mathbf{n}, \mathbf{w}, \mathbf{G}; \boldsymbol{\Theta}_{\mathbf{q}}(k)), \tag{12}$$

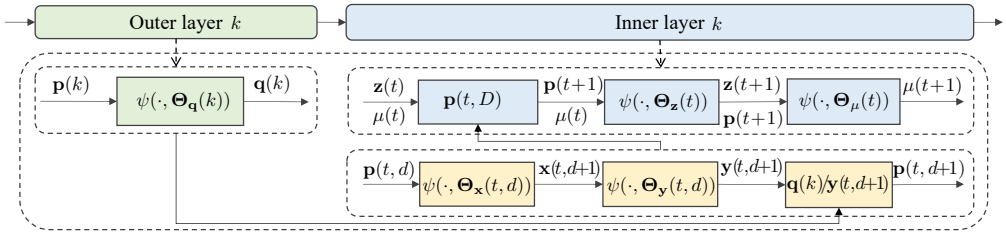



**Figure 1: The architecture of neural sum rate maximization for solving (1). In each outer layer block, the function $\mathbf{q}(k)$ learns the weights $\mathbf{w}^\top \boldsymbol{\alpha}(k)$ for the respective inner layer block. The inner layer block comprises $T$ layers that unroll the iterations of ADMM to solve (6), with $D$ layers dedicated to solving the subproblem of $\mathbf{p}(t+1)$ within these $T$ layers.**

where the function $\boldsymbol{\psi}(\,\cdot\,; \mathbf{\Theta_q}(k))$ is a neural network designed with linear layers and non-linear activation functions. The inputs are $\mathbf{p}(k)$, $\mathbf{n}$ $\mathbf{w}$ and the vectorization of $\mathbf{G}$. The $\mathbf{\Theta_q}(k)$ is the learning parameter set.

We now explore unrolling the inner loops in Algorithm 1 containing the ADMM iterations. There are three key iterative steps of solving the subproblems (8) within the ADMM framework. To update $\mathbf{p}(t+1)$, we employ neural networks to unroll the iteration in (9) into $D$ layers, aiming to reach the fixed point of (9) while keeping the number of layers fixed. This process is outlined as:

$$\mathbf{x}(t, d{+}1) = \boldsymbol{\psi}\left(\mathrm{G}\mathrm{diag}(\mathbf{p}(t, d)), \mathbf{n}; \mathbf{\Theta_x}(t, d)\right), \tag{13}$$

$$\mathbf{y}(t, d{+}1) = \boldsymbol{\psi}\left(\beta(\mathbf{p}(t, d), \mathbf{z}(t), \boldsymbol{\mu}(t)), \mathrm{G}^T \mathrm{diag}\left(\frac{\mathbf{w}}{\mathbf{x}(t, d{+}1)}\right); \mathbf{\Theta_y}(t, d)\right), \tag{14}$$

$$\mathbf{p}(t, d{+}1) = \frac{\mathbf{q}(k)}{\mathbf{y}(t, d{+}1)}, \tag{15}$$

where the function $\boldsymbol{\psi}(\,\cdot\,; \mathbf{\Theta_x}(t, d))$ and $\boldsymbol{\psi}(\,\cdot\,; \mathbf{\Theta_y}(t, d))$ are also neural network models. The learning parameters in (13) and (14) are $\mathbf{\Theta_x}(t, d)$ and $\mathbf{\Theta_y}(t, d)$, respectively. The inputs of (13) are the vectorization of $\mathrm{G}\mathrm{diag}(\mathbf{p}(t, d))$ and $\mathbf{n}$. The inputs of (14) are $\beta(\mathbf{p}(t, d), \mathbf{z}(t), \boldsymbol{\mu}(t))$ and the vectorization of $\mathrm{G}^T \mathrm{diag}\left(\frac{\mathbf{w}}{\mathbf{x}(t,d+1)}\right)$. The $\mathbf{p}(t, D)$, which is the output of $D$ layers of (13) to (15), is utilized to update $\mathbf{p}(t+1)$.

To update the subproblem $\mathbf{z}(t+1)$, we use $\mathbf{p}(t+1)$ from (15), $\boldsymbol{\mu}(t)$, and $\bar{P}$ as inputs to the neural network. The output $\mathbf{z}(t+1)$ is defined as:

$$\mathbf{z}(t+1) = \boldsymbol{\psi}(\mathbf{p}(t+1), \boldsymbol{\mu}(t), \bar{P}; \mathbf{\Theta_z}(t)), \tag{16}$$

where $\mathbf{\Theta_z}(t)$ represents the learning parameter set specific to the $t$-th stage for $\mathbf{z}(t+1)$. Similarly, to update the subproblem $\boldsymbol{\mu}(t+1)$ using neural network, we use $\mathbf{p}(t+1)$, $\mathbf{z}(t+1)$, and $\boldsymbol{\mu}(t)$ as inputs. The output of this layer $\boldsymbol{\mu}(t+1)$ is defined as:

$$\boldsymbol{\mu}(t+1) = \boldsymbol{\psi}(\mathbf{p}(t+1), \mathbf{z}(t+1), \boldsymbol{\mu}(t); \mathbf{\Theta}_{\boldsymbol{\mu}}(t)), \tag{17}$$

where $\mathbf{\Theta}_{\boldsymbol{\mu}}(t)$ represents the learning parameter set.

The architecture of the neural sum rate maximization to solve (1) is outlined in Algorithm 2. Note that this architecture comprises $K \times T \times D$ layers. The $K$ outer layers learn $\mathbf{w}^T \boldsymbol{\alpha}(k)$ to approximate the convex subproblems in (1). The $T$ inner layers unroll the iterations for solving subproblems (8) in the ADMM, with the iteration for updating $\mathbf{p}(t+1)$ in (9) being unrolled into $D$ layers. Within the $D$ layers of (13)-(15), $\mathbf{x}(t)$ learns the step $\sum_{j \neq l} G_{lj} p_j(t-1) + n_j$, while $\mathbf{y}(t)$ learns the step $\beta(p_l(t, d) - z_l(t) + \mu_l(t)) + \sum_{i \neq l} \frac{w_i G_{il}}{\sum_{j \neq i} G_{ij} p_j(t,d) + n_i}$ in (9). Our approach, inspired by works such as [7, 9], leverages algorithm unrolling techniques by making specific parameters of the original iterative algorithm learnable. Specifically, from (13) to (17), we make the addition operation trainable by using a learnable neural network, i.e., making the matrix of ones trainable.

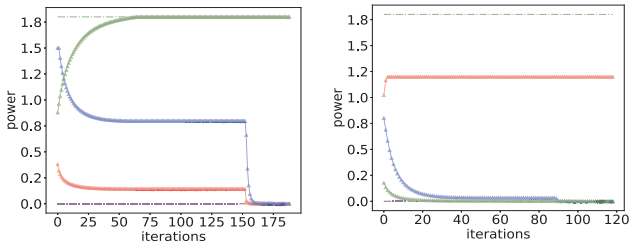
**Algorithm 2:** Neural Sum Rate Maximization

**Initialize**: The number of layers $K$, $T$ and $D$.
**Input:** The system parameters $\mathbf{G}$, $\mathbf{n}$ $\mathbf{w}$ and $\bar{P}$ and initialized $p_l(0) = \frac{1}{L}\bar{P}$ for all $l$.
**Output:** The optimal solution $\mathbf{p}^*$ of (1).
**for** $k = 0$ **to** $K - 1$ **do**
  $\mathbf{q}(k) = \boldsymbol{\psi}(\mathbf{p}(k), \mathbf{n}, \mathbf{w}, \mathbf{G}; \mathbf{\Theta_q}(k))$;
  **for** $t = 0$ **to** $T - 1$ **do**
    **for** $d = 0$ **to** $D - 1$ **do**
      Update $\mathbf{x}(t, d+1)$ with (13);
      Update $\mathbf{y}(t, d+1)$ with (14);
      Update $\mathbf{p}(t, d+1)$ with (15);
    **end**
    $\mathbf{p}(t+1) \leftarrow \mathbf{p}(t, D)$;
    $\mathbf{z}(t+1) = \boldsymbol{\psi}(\mathbf{p}(t+1), \boldsymbol{\mu}(t), \bar{P}; \mathbf{\Theta_z}(t))$;
    $\boldsymbol{\mu}(t+1) = \boldsymbol{\psi}(\mathbf{p}(t+1), \mathbf{z}(t+1), \boldsymbol{\mu}(t); \mathbf{\Theta}_{\boldsymbol{\mu}}(t))$.
  **end**
  $\mathbf{p}(k{+}1) \leftarrow \mathbf{p}(T)$.
**end**

## 3.2 Training Process

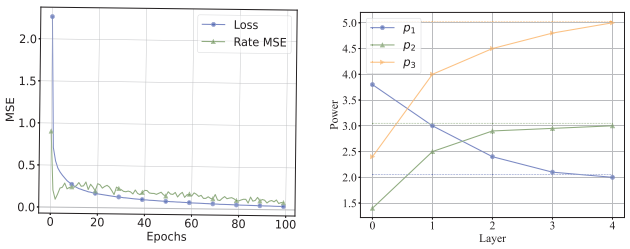
Now let us consider training the parameters in the neural sum rate maximization in a supervised manner. Denote the Algorithm 2 with respect to the learning parameter by $\mathbf{\Phi}(\mathbf{\Theta})$, where $\mathbf{\Theta}$ is the learning parameter set with $\mathbf{\Theta} = \left\{\mathbf{\Theta_q}(k), \mathbf{\Theta_x}(t, d), \mathbf{\Theta_y}(t, d), \mathbf{\Theta_z}(t), \mathbf{\Theta}_{\boldsymbol{\mu}}(t)\right\}$. Suppose we have a dataset $\mathcal{D}$ consisting of $|\mathcal{D}|$ training samples for (1). The loss function between the output of Algorithm 2 with the ground truths $\mathbf{p}^*$ is defined as

$$\mathbf{E}(\mathbf{\Theta}) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{G}, \mathbf{w}, \bar{\mathbf{p}}, \mathbf{p}^*) \in \mathcal{D}} \|\mathbf{\Phi}(\mathbf{\Theta}) - \mathbf{p}^*\|_2^2. \tag{18}$$




(a) (b)

**Figure 2: Illustration of the exact optimal solution (a) and the sub-optimal solution (b) obtained from Algorithm 1 with different initial points.**




(a) (b)

**Figure 3: Illustration of the training loss and MSE of the predicted sum rate across the training epochs (a) and the evolution of power across the number of layers of $K$ (b) for the neural sum rate maximization.**

Our goal is to find the optimal $\boldsymbol{\Theta}^*$ that minimizes the $\mathbf{E}(\boldsymbol{\Theta})$ with the given $\mathcal{D}$. To build labeled training data, we can construct problem instances that adhere to the existence of the quasi-inverse matrix for the system configuration of the fast algorithm in [18]. Then we can utilize gradient descent to iteratively adjust the learning parameters, minimizing the loss function and enhancing the model's performance.

## 4 PERFORMANCE EVALUATION

In this section, we present simulation results showcasing the performance of our neural sum rate maximization for the downlink wireless system. The data and code for the numerical experiments can be accessed at https://github.com/convexsoft/Neural-Sum-Rate-Maximization.

### 4.1 Performance of Algorithm 1

Consider a multi-access wireless network that serves $L = 3$ users. The downlink transmit powers are denoted by $p_1$, $p_2$, and $p_3$. The maximum power allowable is $\bar{P} = 1.85$ mW, and the noise power at each user is $\mathbf{n} = [0.05, 0.05, 0.05]^\top$ mW. We configure the channel gain as $\mathbf{G} = [1.53, 0.19, 0.01; 0.01, 1.52, 0.02; 0.16, 0.53, 1.78]$ and the weights for the weighted sum rate objective as $\mathbf{w} = [1/3, 1/3, 1/3]^\top$. Next, we utilize Algorithm 1 to maximize the sum rate in the system. We set the penalty parameter $\beta = 10e^{-3}$. For an initial power setting of $\mathbf{p} = [0.90, 1.50, 0.35]^\top$ mW, Figure 2 (a) illustrates the evolution of transmit powers (depicted in solid curves) for the 3 users, indicating that the power evolution converges towards the optimal solution (depicted in dashed lines). When the initial power is selected as $\mathbf{p} = [0.16, 0.85, 1.00]^\top$ mW, Figure 2 (b) demonstrates the evolution of transmit powers (depicted in solid curves). The results suggest that the power evolution may not always converge to the optimal solution unless carefully chosen initial points are used.

### 4.2 Performance of Algorithm 2

To configure the neural sum rate maximization in Algorithm 2, we set $K = T = D = 4$ and configure each neural network with three hidden layers, each consisting of 128 neurons and utilizing the RELU activation function. The learning rate is $1 \times 10^{-3}$, weight decay is $1 \times 10^{-4}$, and the training is restricted to a maximum of 100 epochs using the loss function defined in (18). Figure 3 (a) displays the training loss and mean squared error (MSE) of the predicted sum rate across epochs, while Figure 3 (b) presents the evolution of power across the layer of $K$. The results indicate that the neural sum rate maximization model exhibits strong performance. This suggests that solutions to the problem can be achieved with a minimal number of iterative steps, thereby greatly improving the efficiency of problem-solving.

### 4.3 Performance Comparison

Now we compare the performance of the neural sum rate maximization in Algorithm 2 with established baselines, which include optimization-theoretic-based algorithms such as the fast algorithm (FA) in [18], the WMMSE algorithm in [23], and the SCA algorithm in [4]. Additionally, we consider modern learning-based baselines utilizing machine learning techniques like the TD3 algorithm in [11] and the GNN algorithm in [8]. To assess the models, we evaluate them across three metrics: power accuracy (Power Acc.), training time ($T_{\text{Training}}$), and testing time ($T_{\text{Testing}}$), as detailed in TABLE 1. We conduct the comparative analysis of neural sum rate maximization with the aforementioned baselines across varying user scales ($L = 4, L = 16$, and $L = 64$) in wireless networks. We utilize problem instances that do not meet the specified conditions for the FA as outlined in [18], thereby making the FA infeasible for these particular instances. Results in TABLE 1 reveal that our proposed Algorithm 1 achieves comparable power accuracy to SCA and WMMSE, with improved efficiency in running time compared to SCA. Furthermore, the neural sum rate maximization approach in Algorithm 2 demonstrates significant advancements in both accuracy and efficiency compared to optimization-based baselines. Compared to modern learning-based methods, the neural sum rate maximization requires less training time than TD3, making it easier to train. Despite similar training times to GNN, the neural sum rate maximization achieves higher accuracy, leading to superior overall performance.

## 5 CONCLUSIONS

This paper introduces *Neural Sum Rate Maximization* for solving sum rate maximization problems in downlink multiple access scenarios. Combining the theoretical optimization methods with neural network-based algorithm unrolling, we create trainable neural network layers for algorithm development in AI-native wireless networks. Our approach utilizes mathematical structures in the

| System Scales | L=4 | | | L=16 | | | L=64 | | |
|---|---|---|---|---|---|---|---|---|---|
| Metrics | Power Acc. | $T_{\text{Training}}$ | $T_{\text{Testing}}$ | Power Acc. | $T_{\text{Training}}$ | $T_{\text{Testing}}$ | Power Acc. | $T_{\text{Training}}$ | $T_{\text{Testing}}$ |
| FA [18] | - | - | - | - | - | - | - | - | - |
| WMMSE [23] | 0.9395 | - | 4.96 | 0.9005 | - | 22.83 | 0.9083 | - | 59.74 |
| SCA [4] | 0.9318 | - | 4.56 | 0.9192 | - | 21.32 | 0.9034 | - | 64.76 |
| Algorithm 1 | 0.9415 | - | 4.70 | 0.9249 | - | 19.66 | 0.9040 | - | 67.60 |
| Algorithm 2 | 0.9836 | 18.34 | 1.30 | 0.9943 | 68.55 | 2.68 | 0.9854 | 1326.7 | 8.02 |
| TD3 [11] | 0.9633 | 48.43 | 2.21 | 0.9721 | 153.30 | 3.61 | 0.9561 | 4546.4 | 14.74 |
| GNN [8] | 0.9492 | 19.17 | 1.81 | 0.9466 | 71.60 | 2.87 | 0.9295 | 1766.5 | 8.99 |

**Table 1: Performance comparison between the neural sum rate maximization and other baselines.**

alternating direction method of multipliers and the standard interference function frameworks, enhancing efficiency through algorithm unrolling and learning from data. Numerical experiments demonstrate improved efficiency, performance, and interpretability, making it valuable for resource-constrained optimization in future AI-native wireless networks.

## ACKNOWLEDGMENTS

This work is supported in part by the Singapore Ministry of Education Academic Research Fund (RG91/22 and NTU startup) and in part by the Science, Technology, and Innovation Commission of Shenzhen Municipality under Project JCYJ20170818094955771.

## REFERENCES

[1] B. Amos. 2023. Tutorial on amortized optimization. *Foundations and Trends in Machine Learning* 16, 5 (2023), 592–732.
[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning* 3, 1 (2011), 1–122.
[3] S. Chen and C. W. Tan. 2023. Neural sum rate maximization with deep unrolling. In *Proceedings of the 2023 IEEE Global Communications Conference*.
[4] M. Chiang, C. W. Tan, D. P. Palomar, D. O'neill, and D. Julian. 2007. Power control by geometric programming. *IEEE Transactions on Wireless Communications* 6, 7 (2007), 2640–2651.
[5] S. S. Christensen, R. Agarwal, E. De Carvalho, and J. M. Cioffi. 2008. Weighted sum-rate maximization using weighted MMSE for MIMO-BC beamforming design. *IEEE Transactions on Wireless Communications* 7, 12 (2008), 4792–4799.
[6] P. L. Combettes and J.-C. Pesquet. 2021. Fixed point strategies in data science. *IEEE Transactions on Signal Processing* 69 (2021), 3878–3905.
[7] K. Gregor and Y. LeCun. 2010. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning*.
[8] M. Lee, G. Yu, and G. Y. Li. 2020. Graph embedding-based wireless link scheduling with few training samples. *IEEE Transactions on Wireless Communications* 20, 4 (2020), 2282–2294.
[9] V. Monga, Y. Li, and Y. C. Eldar. 2021. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine* 38, 2 (2021), 18–44.
[10] T. O'Shea and J. Hoydis. 2017. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking* 3, 4 (2017), 563–575.
[11] M. Rahmani, M. Bashar, M. J. Dehghani, A. Akbari, P. Xiao, R. Tafazolli, and M. Debbah. 2023. Deep reinforcement learning-based sum rate fairness trade-off for cell-free mMIMO. *IEEE Transactions on Vehicular Technology* 72, 5 (2023), 6039–6055.
[12] R. Sambharya, G. Hall, B. Amos, and B. Stellato. 2023. Learning to warm-start fixed-point optimization algorithms. *arXiv preprint arXiv:2309.07835* (2023).
[13] L. Schynol and M. Pesavento. 2023. Coordinated sum-rate maximization in multicell MU-MIMO with deep unrolling. *IEEE Journal on Selected Areas in Communications* 41, 4 (2023), 1120–1134.
[14] K. Shen and W. Yu. 2018. Fractional programming for communication systems-Part I: Power control and beamforming. *IEEE Transactions on Signal Processing* 66, 10 (2018), 2616–2630.
[15] J. Song, C. Häger, J. Schröder, T. O'Shea, E. Agrell, and H. Wymeersch. 2022. Benchmarking and interpreting end-to-end learning of MIMO and multi-user communication. *IEEE Transactions on Wireless Communications* 21, 9 (2022), 7287–7298.
[16] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos. 2018. Learning to optimize: Training deep neural networks for interference management. *IEEE Transactions on Signal Processing* 66, 20 (2018), 5438–5453.
[17] Y. Sun, P. Babu, and D. P. Palomar. 2016. Majorization-minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing* 65, 3 (2016), 794–816.
[18] C. W. Tan. 2015. Wireless network optimization by Perron-Frobenius theory. *Foundations and Trends® in Networking* 9, 2-3 (2015), 107–218.
[19] C. W. Tan, S. Friedland, and S. H. Low. 2011. Nonnegative matrix inequalities and their application to nonconvex power control optimization. *SIAM Journal on Matrix Analysis and Applications* 32, 3 (2011), 1030–1055.
[20] C. W. Tan, S. Friedland, and S. H. Low. 2011. Spectrum management in multiuser cognitive wireless networks: Optimality and algorithm. *IEEE Journal on Selected Areas in Communications* 29, 2 (2011), 421–430.
[21] S. Venkataraman and B. Amos. 2021. Neural fixed-point acceleration for convex optimization. *arXiv preprint arXiv:2107.10254* (2021).
[22] R. D. Yates. 1995. A framework for uplink power control in cellular radio systems. *IEEE Journal on Selected Areas in Communications* 13, 7 (1995), 1341–1347.
[23] M. Zaher, Ö. T. Demir, E. Björnson, and M. Petrova. 2022. Learning-based downlink power allocation in cell-free massive MIMO systems. *IEEE Transactions on Wireless Communications* 22, 1 (2022), 174–188.
[24] L. Zheng, Y.-W. P. Hong, C. W. Tan, C.-L. Hsieh, and C.-H. Lee. 2016. Wireless max–min utility fairness with general monotonic constraints by Perron–Frobenius theory. *IEEE Transactions on Information Theory* 62, 12 (2016), 7283–7298.
[25] L. Zheng and C. W. Tan. 2014. Maximizing sum rates in cognitive radio networks: Convex relaxation and global optimization algorithms. *IEEE Journal on Selected Areas in Communications* 32, 3 (2014), 667–680.

# A APPENDIX

## A.1 Proof of Lemma 2.1

Proof. Let $\tilde{\mathbf{p}} = \log \mathbf{p}$. By taking the derivative of the objective function of the subproblem for $\tilde{\mathbf{p}}$ in the ADMM, and setting it to zero, we obtain the iteration (9). Let

$$I_l(\mathbf{p}) = \frac{\mathbf{w}^\top \boldsymbol{\alpha}_l(k)}{\beta(p_l(t,d) - z_l(t) + \mu_l(t)) + \sum_{i \neq l} \dfrac{w_i G_{il}}{\sum_{j \neq i} G_{ij} p_j(t,d) + n_i}}.$$

It is obvious that for any $\mathbf{0} \leq \mathbf{p}' \leq \mathbf{p}''$, we have $\mathbf{I}(\mathbf{p}') \leq \mathbf{I}(\mathbf{p}'')$. Let $\eta > 1$. When $I_l(\eta\mathbf{p}) < \bar{p}_l$, then $\eta I_l(\mathbf{p}) = \eta h_l(\mathbf{p}) > h_l(\eta\mathbf{p}) = I_l(\eta\mathbf{p})$. When $I_l(\eta\mathbf{p}) = \bar{p}_l$ and $I_l(\mathbf{p}) = h_l(\mathbf{p}) < \bar{p}_l$, since $I_l(\eta\mathbf{p}) < h_l(\eta\mathbf{p})$, then $\eta I_l(\mathbf{p}) = \eta h_l(\mathbf{p}) > h_l(\eta\mathbf{p}) > I_l(\eta\mathbf{p})$. When $I_l(\eta\mathbf{p}) = \bar{p}_l$ and $I_l(\mathbf{p}) = \bar{p}_l$, then $\eta I_l(\mathbf{p}) = \eta\bar{p}_l > \bar{p}_l = I_l(\eta\mathbf{p})$. Therefore, for any $\eta > 1$ and $\mathbf{p} \geq \mathbf{0}$, we have $\mathbf{I}(\eta\mathbf{p}) < \eta\mathbf{I}(\mathbf{p})$, which means $\mathbf{I}(\mathbf{p})$ is a standard interference function [22]. When $\mathbf{p} \to +\infty$, $\mathbf{I}(\mathbf{p}) \leq \bar{\mathbf{p}}$, i.e., $\mathbf{I}(\mathbf{p}) < \mathbf{p}$. According to the standard interference function properties in [22], $\mathbf{I}(\mathbf{p})$, the solution of $\mathbf{p}(t+1)$, i.e., the fixed point of $\mathbf{I}(\mathbf{p})$ can be obtained by the iteration (9). □

## A.2 Proof of Lemma 2.2

Proof. When $\sum_{l=1}^{L} z_l(t+1) \leq \bar{P}$, then $\mathbf{z}(t+1) = \operatorname{argmin}_{\mathbf{z}} \frac{\beta}{2} \|\mathbf{p}(t+1) - \mathbf{z} + \boldsymbol{\mu}(t)\|_2^2$. Thus, $\mathbf{z}(t+1) = \mathbf{p}(t+1) + \boldsymbol{\mu}(t)$. When $\sum_{l=1}^{L} z_l(t+1) > \bar{P}$, then $\mathbf{z}(t+1)$ is the projection of $\mathbf{p}(t+1) + \boldsymbol{\mu}(t)$ on the hyperplane $\sum_{l=1}^{L} z_l(t+1) = \bar{P}$. Let $t = \frac{1}{L}\left(\sum_{l=1}^{L} (p_l(t+1) + \mu_l(t)) - \bar{P}\right)$. Then we have $\mathbf{z}(t+1) = \mathbf{p}(t+1) + \boldsymbol{\mu}(t) - t\mathbf{1}$. In conclusion, we can derive that $z_l(t+1) = p_l(t+1) + \mu_l(t) - \frac{1}{L}\left(\sum_{l=1}^{L} (p_l(t+1) + \mu_l(t)) - \bar{P}\right)_+$. □